2026-1 세미나

Toward real-world centric Text2SQL task

NLP&AI 강명훈

Natural Language Processing & Artificial Intelligence

고려대학교 KOREA UNIVERSITY

What is Text2SQL?

- **Leveraging structural reasoning capability with LM**

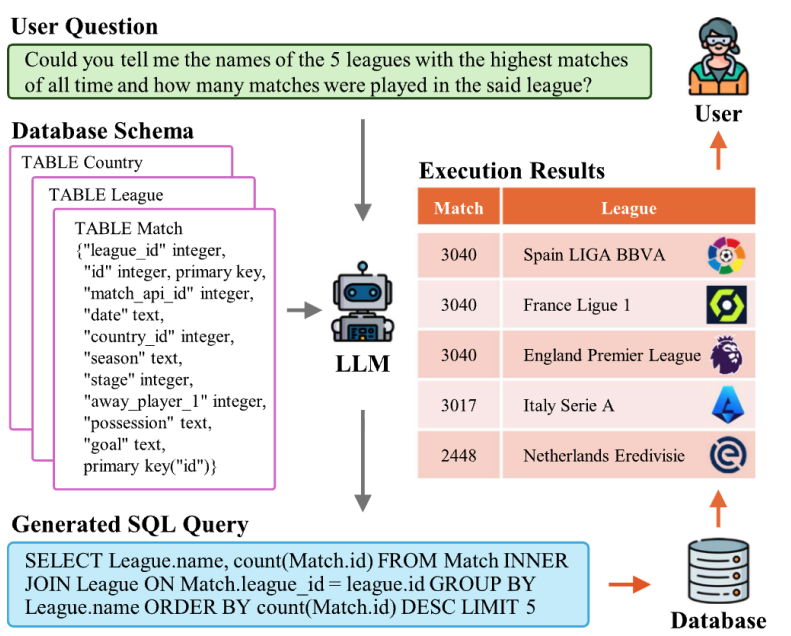


Fig. 1. An example of LLM-based text-to-SQL is selected from the BIRD [1] dataset. A user asks a question about football leagues. The LLM takes this question along with the schema of the corresponding database as input and generates an SQL query as output. The generated SQL query can be executed in the database, retrieves the content *"The 5 leagues with the highest matches"*, providing the answer to the user's question.

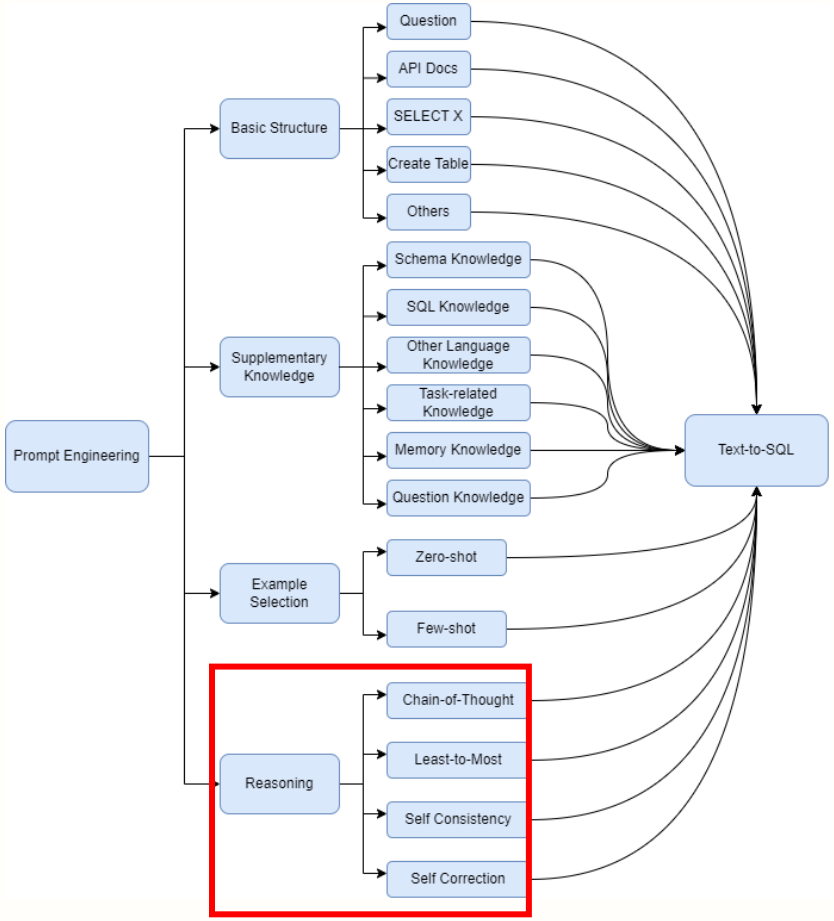
- Text2SQL (NL2SQL): 사용자가 자연어로 물어본 질의 (Text)의 맥락과 의도에 부합하는 근거 답변을 구조화된 자료 DB로부터 추출할 수 있는 SQL 질의문을 생성하는 과업

- 정확한 Text2SQL 과업을 수행하기 위해서는 다음의 challenge (sub-task) 수행이 필수적
  - Linguistic Complexity and Ambiguity: 자연어로 표현된 사용자 질의 속 **맥락과 의도를 정확하게 파악하는 능력** 필요
  - Schema Understanding and Representation: 질의에 부합하는 답변을 생성하기 위해 **참조하는 DB의 구조, 내용을 이해**할 수 있어야 함. 이러한 이해는 단일 Table 구조, 내용 이해부터 Table간 관계 파악 등을 포함
  - Rare and Complex SQL Operations: 사용자의 질의 해결을 위해 여러 sub-query를 생성하는 Nested-query, Window function 등과 같이 복잡한 SQL 구문 생성 능력
  - Cross-Domain Generalization: System이 학습하거나 추론하는 DB의 domain에 관계없이 일반화된 성능을 발휘할 수 있어야 함

Hong, Z., Yuan, Z., Zhang, Q., Chen, H., Dong, J., Huang, F., & Huang, X. (2025). Next-generation database interfaces: A survey of llm-based text-to-sql. IEEE Transactions on Knowledge and Data Engineering.

What is Text2SQL?

- **How to resolve these challenges for successful Text2SQL performance?**



- TextSQL에서 요구되는 4가지 challenge를 달성하기 위한 다양한 methodology가 존재할 수 있음
  - Basic Structure: SQL과 관련된 기초 지식을 주입 및 전달하는 방법
  - Supplementary Knowledge: SQL과 관련된 보조 지식을 주입 및 전달하여 SQL 구문을 정확하게 생성하게 하는 방법
  - Example Selection: ICL 방식으로 SQL 구문을 생성함에 있어서 일반화된 생성을 도모하는 example sampling 방법
  - Reasoning: 정확한 SQL 구문 생성을 위한 fine-grained 추론 과정 정립에 관한 방법

- 최근 Reasoning trend와 맞물려 Text2SQL 과업에서 LLM, LRM을 활용하여 어떻게 정확한 SQL 구문을 생성할 것인가와 관련된 연구가 활발하게 진행중

Shi, L., Tang, Z., Zhang, N., Zhang, X., & Yang, Z. (2025). A survey on employing large language models for text-to-sql tasks. ACM Computing Surveys, 58(2), 1-37.

How recent trends are embedded in Text2SQL

LEARNAT: LEARNING NL2SQL WITH AST-GUIDED TASK DECOMPOSITION FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

Text2SQL task에서 LLM의 sub-task decomposition 능력을 향상하기 위한 합성 데이터 구축 및 특화 학습 방법론 제안

BIRD-INTERACT: RE-IMAGINING TEXT-TO-SQL EVALUATION VIA LENS OF DYNAMIC INTERACTIONS

Anonymous authors
Paper under double-blind review

Multi-turn conversation, Agentic 상황에서의 Text2SQL interaction을 simulate, evaluate할 수 있는 평가 프레임워크 제안

LEARNAT: LEARNING NL2SQL WITH AST-GUIDED TASK DECOMPOSITION FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ICLR 2026 poster

Score: 6 / 8 / 8

LearNAT

- **Motivations: Task decomposition is a critical bottleneck for applying LLM to Text2SQL**

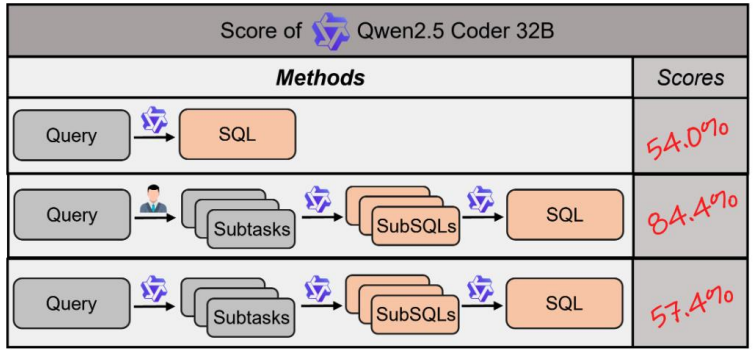


Figure 6: A preliminary experiment was conducted. We randomly selected 500 cases from the BIRD Train dataset and employed QWen-2.5-Coder to perform the NL2SQL task. The experimental results indicate that *enhancing the LLM's task decomposition ability is crucial for improving its performance on NL2SQL tasks.*

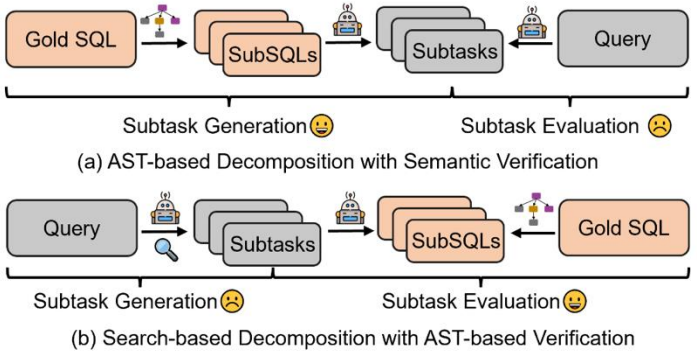


Figure 7: AST-based Decomposition *vs.* Search-based Decomposition

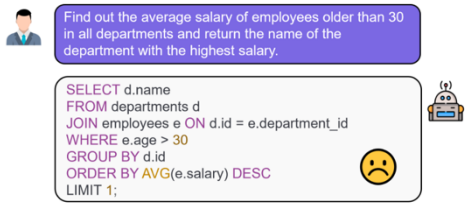
Table 5: Experimental results on evaluating subtask correctness using language models.

Models	Accuracy		
	Correct (227)	Error (273)	Total (500)
sentence-transformer	92.1	9.2	46.8
GLM-4	45.4	20.5	36.0

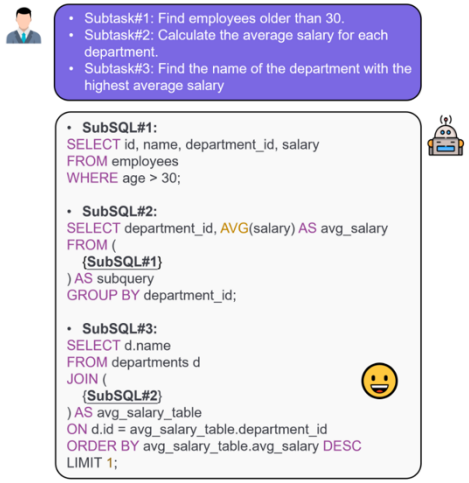
- LLM에게 Oracle sub-task decomposition을 줄 경우 SQL 성능 향상이 획기적으로 이뤄지는 점을 발견
- 그러나 LLM Sub-task Decomposition의 quality가 떨어지고 분해된 sub-task의 LLM 방식 검증 정확성이 매우 떨어지는 상황

LearNAT

- **Motivations: Task decomposition is a critical bottleneck for applying LLM to Text2SQL**



(a) LLM on complex NL2SQL task.



(b) LLM on multiple simple NL2SQL subtasks.

Figure 4: (a) illustrates the LLM directly solving a complex NL2SQL task, resulting in an incorrect output. (b) shows the LLM solving multiple decomposed simple NL2SQL subtasks from the same task in (a), resulting in a correct output. This motivates our approach to *enhancing the LLM's ability to decompose complex tasks*, thereby improving its performance on challenging NL2SQL queries.

LearNAT – Decomposition Synthesis Procedure

- **Overview for Decomposition Synthesis Procedure**

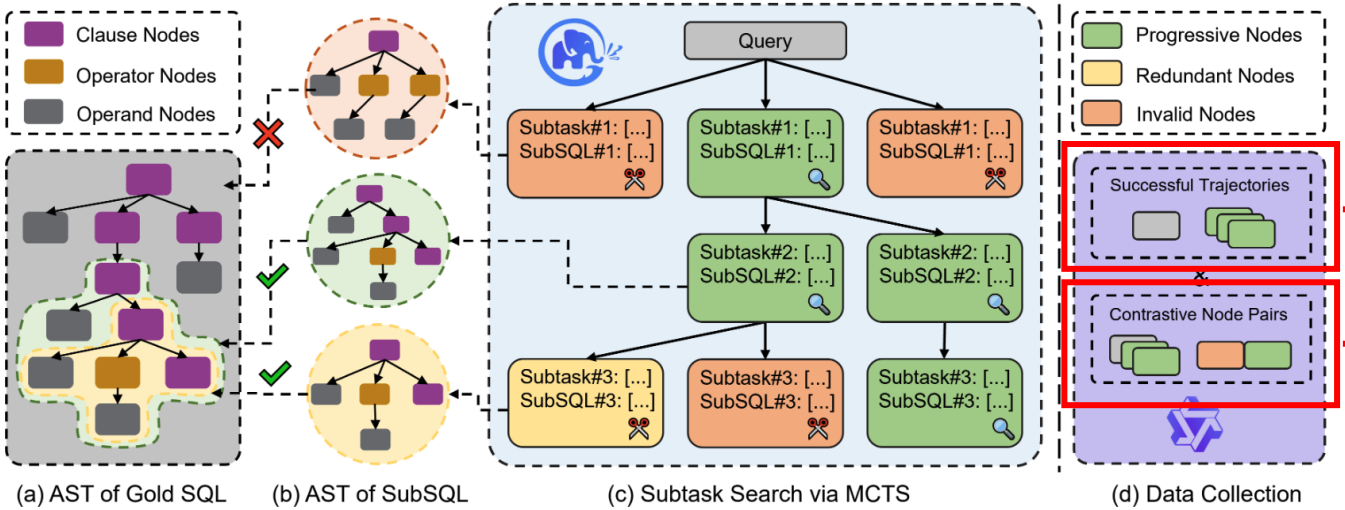


Figure 2: Framework of the *Decomposition Synthesis Procedure*. (c) illustrates how the LLM, combined with MCTS, performs next-step prediction to synthesize subtasks of complex NL2SQL tasks. (b) presents the AST of the SQL statements corresponding to each synthesized subtask in (c). (a) shows the AST of the Gold SQL for the complex NL2SQL task, which guides the MCTS in (c) to perform more efficient search, including pruning and node reward estimation. (d) depicts the data collected by `LearNAT` during the *Decomposition Synthesis Procedure*, comprising successful trajectories data for supervised fine-tuning and step-wise contrastive node pairs data for preference learning. Under the default settings of `LearNAT`, GLM-4-Plus is used to synthesize decomposition data, and the Qwen2.5-Coder model is fine-tuned.

LLM의 sub-task Decomposition 능력 향상을 위한 Verifiable Decomposition dataset 구축 방법을 제안

SFT 용도

Preference learning 용도

GLM-4-Plus로 Gold-query에 대하여 MCTS 개념과 AST parsing 결과를 활용한 verifiable decomposition 실시

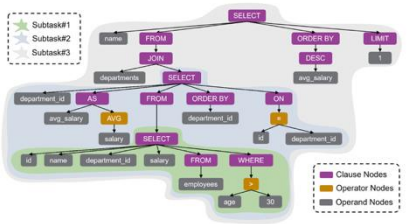


Figure 5: The abstract syntax tree (AST) of the given case in Fig. 4. Each simple NL2SQL subtask in Fig. 4 corresponds to a subtree within the AST. Clause nodes, operator nodes and operand nodes were defined in Sec. B.

LearNAT – Decomposition Synthesis Procedure

- **Problem Definition**

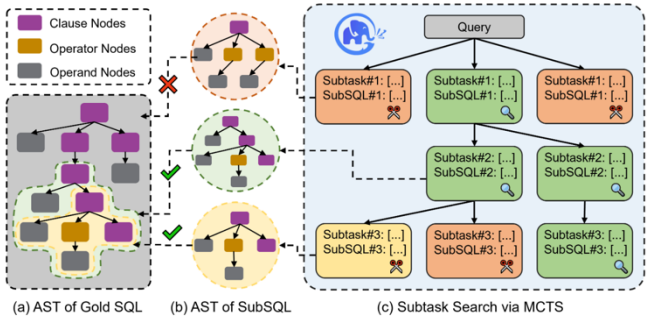
MCTS 방식으로 Decomposition을 실시 -> Tree search problem
sub-task decomposition을 MCTS 개념을 적용하면 state는 i번째 sub-task, action은 decomposition에 해당
$s_i = \{q_i, y_i, \mathcal{AT}(y_i), \mathcal{AT}_{\text{sum}}(y_i), R(s_i)\}$는 i번째 query decomposition state

-

Symbol	Description
Natural Language to SQL (NL2SQL)	
Q	Natural language (NL) question
Y	Corresponding SQL query
$\mathcal{DB}$	Database schema
$\mathcal{K}$	External knowledge
Abstract Syntax Trees (AST)	
$\mathcal{AT}(Y) = (\mathcal{N}, \mathcal{E})$	Abstract Syntax Tree, a directed acyclic graph of SQL query Y
$\mathcal{N}$	Set of nodes in AST
$\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$	Set of edges in AST
$n_c \in \mathcal{N}_c$	Clause Nodes
$n_o \in \mathcal{N}_o$	Operator Nodes
$n_v \in \mathcal{N}_v$	Operand Nodes

$$\mathcal{AT}_{\text{sum}}(y_i) = (\mathcal{N}_{\text{sum}}, \mathcal{E}_{\text{sum}}), \quad (1)$$

$$\mathcal{N}_{\text{sum}} = \bigcup_{j=1}^{i} \mathcal{N}(\mathcal{AT}(y_j)), \mathcal{E}_{\text{sum}} = \bigcup_{j=1}^{i} \mathcal{E}(\mathcal{AT}(y_j)). \quad (2)$$



Sub-SQL yi는 AST parsing을 통해서 특정 node로 분류될 수 있음 (node classification)

Progressive Node: $\mathcal{AT}(y_i)$의 sub-tree이면서 $\mathcal{AT}_{\text{sum}}(y_{\text{parent}(i)})$의 sub-tree가 아닌 경우, 즉 최종 SQL 생성에 관여하면서도 누적 sub-query와 구분되는 새로운 sub-query -> **Decomposition 가치 O**

Redundant Node: $\mathcal{AT}(y_i)$의 sub-tree이면서 $\mathcal{AT}_{\text{sum}}(y_{\text{parent}(i)})$의 sub-tree인 경우, 즉 최종 SQL 생성에 관여하나 누적 sub-query에 포함되는 경우 -> **Decomposition 가치 X**

Invalid Node: $\mathcal{AT}(y_i)$의 sub-tree가 아닌 경우, 잘못된 decomposition -> **Decomposition 가치 X**

- Node classification을 통해서 decomposition 과정에 중요한, 중요하지 않은 sub-task를 선정하여 SFT, DPO 학습 label mapping에 활용

LearNAT – Decomposition Synthesis Procedure

- **Problem Definition**
  - 본 논문은 step-level fine-grained sub-task decomposition을 지향. 따라서 각 decomposition step별 정상 수행 여부 확인 및 교정을 위한 reward가 필요
  - 향후 DPO기반 preference learning에서 사용할 win, lose case간의 fine-grained margin을 부여하기 위한 reward estimation을 진행

$$\mathcal{R}(s_i) = \alpha \cdot \text{sim}_{\text{node}}(\mathcal{AT}_{\text{sum}}(y_i), \mathcal{AT}(Y)) + (1-\alpha) \cdot \text{sim}_{\text{struct}}(\mathcal{AT}_{\text{sum}}(y_i), \mathcal{AT}(Y)), \quad (5)$$

$$\text{sim}_{\text{node}}(\mathcal{AT}_1, \mathcal{AT}_2) = \sum_{t \in \{c,o,v\}} w_t \cdot \text{sim}_t(\mathcal{AT}_1, \mathcal{AT}_2), \quad (14)$$

$$\text{sim}_{\text{struct}}(\mathcal{AT}_1, \mathcal{AT}_2) = 1 - \frac{\text{TED}(\mathcal{AT}_1, \mathcal{AT}_2)}{\max(|\mathcal{AT}_1|, |\mathcal{AT}_2|)}, \quad (16)$$

LearNAT – Margin-Aware Reinforcement Learning

- **Warm-up strategy for Foundational Skill Acquisition**

  LLM을 Sub-task decomposition에 특화되도록 초기 학습을 SFT 방식으로 진행

  입력: $(\mathcal{Q}, \mathcal{DB}, \mathcal{K})$ -> original query, DB schema, external knowledge
  출력: $\{(q_1, y_1), \ldots, (q_n, y_n)\}$)

$$\mathcal{L}_{\mathrm{SFT}} = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{t})}\left[\sum_{i=1}^{I} \log p_\theta\left(t_i \mid \boldsymbol{t}_{1:i-1}, \boldsymbol{x}\right)\right], \tag{6}$$

(Q, DB, K, {(q1, y1), · · · , (qn, yn)})

$$p_\theta(\boldsymbol{t} \mid \boldsymbol{x}) = \prod_{i=1}^{I} p_\theta\left(t_i \mid \boldsymbol{t}_{<i}, \boldsymbol{x}\right)$$

LearNAT – Margin-Aware Reinforcement Learning

- **DPO with AST Margin**

DPO 기반의 preference learning을 수행하여

입력: $(\mathcal{Q}, \mathcal{DB}, \mathcal{K}, (q_1, y_1), \ldots, (q_{i-1}, y_{i-1}))$ -> original query, DB schema, external knowledge, i-1번째 까지의 sub-task decomposition 결과

Win: $((q_i^w,\ y_i^w))$

Lose: $((q_i^l,\ y_i^l))$

$$\mathcal{L}_{\mathrm{DPO}}(\pi_\theta; \pi_{\mathrm{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma\left(\hat{r}_\theta(x, y_w) - \hat{r}_\theta(x, y_l)\right)\right]$$

$$\hat{r}_\theta(x, y) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)}. \qquad (13)$$

그러나 DPO 방식의 preference learning으로는 **step-wise sub-task decomposition을 학습하기 어려움**

Fine-grained sub-task decomposition을 위한 **AST margin을 DPO loss항에 추가하여 Preference learning 실시**

$$\mathcal{L}_{\mathrm{MDPO}}(\pi_\theta; \pi_{\mathrm{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma\left(\hat{r}_\theta(x, y_w) - \hat{r}_\theta(x, y_l) - \triangle_r\right)\right], \qquad (8)$$

$$\text{where } \triangle_r = \mathrm{margin}((q_i^w, y_i^w), (q_i^l, y_i^l)) \qquad \mathrm{margin}((q_i^w, y_i^w), (q_i^l, y_i^l)) = \mathcal{R}(s_i^w) - \mathcal{R}(s_i^l).$$

- Wine, lose간의 reward margin을 loss에 추가함으로써 win case가 lose대비 **얼마나 더 선호될 수 있는지**에 관한 signal을 줄 수 있음
- 최종 sub-query 생성까지 경우로 win, lose dataset을 설정하지 **않고 중간 단계 까지의 decomposition 수행 결과를 바탕으로도 학습을 진행**할 수 있음

LearNAT – Experimental Setups

- **Experimental dataset**

Table 7: Statistics for NL2SQL benchmarks.

Benchmarks	#Queries				
BIRD-train	~~9,428~~ 9,003				
BIRD-dev	Simple 925	Moderate 465	Challenging 144		Total 1,534
Spider-dev	Easy 248	Medium 446	Hard 174	Extra Hard 166	Total 1,034

{'question_id': 0,
'db_id': 'california_schools',
'question': 'What is the highest eligible free rate for K-12 students in the schools in Alameda County?',
'evidence': 'Eligible free rate for K-12 = `Free Meal Count (K-12)` / `Enrollment (K-12)`',
'SQL': "SELECT `Free Meal Count (K-12)` / `Enrollment (K-12)` FROM frpm WHERE `County Name` = 'Alameda' ORDER BY (CAST(`Free Meal Count (K-12)` AS REAL) / `Enrollment (K-12)`) DESC LIMIT 1",
'difficulty': 'simple'}

LearNAT – Experimental Setups

- **Implementation details**

- **Metric**

SQL Query generation Evaluation Metric

- **Execution Accuracy (EX)**

$$\text{EX} = \frac{\sum_{i=1}^{N} \mathbb{1}(\hat{V}_i, V_i)}{N}$$

$$\mathbb{1}(\hat{V}_i, V_i) = \begin{cases} 1 & \text{if} \quad \hat{V}_i = V_i \\ 0 & \text{if} \quad \hat{V}_i \neq V_i \end{cases}$$

F.1 IMPLEMENTATION DETAILS.

We employ GLM-4-Plus[5] as the primary model for synthesizing decomposition data and fine-tune the model on Qwen2.5-Coder (Yang et al., 2024a), including its 7B, 14B, and 32B versions. We used the PyTorch library to implement all the algorithms based on the open-source HuggingFace transformers (Wolf et al., 2019) and LLaMA-Factory (Zheng et al., 2024). The experiments are conducted on 8×A100 GPUs. During the SFT stage, we utilize the AdamW optimizer with a learning rate of 2e-5 and a cosine warmup scheduler over three epochs. For DPO training, the Adam optimizer is used with a learning rate of 2e-6, and the β parameter is set to 0.2, in accordance with the original DPO configuration. In Eq. 14, we assign equal weights to all three nodes, i.e., $w_c = w_o = w_v = 0.33$. Based on our experimental observations F.12, we set $\alpha = 0.75$ in Eq. 5. During inference, we strictly follow the evaluation protocol provided by DAIL-SQL (Gao et al., 2024) (the Without Voting setting). The protocol provides a complete set of prompts to better structure the instructions, user queries, and database schema information, enabling the LLM to generate a single response from which the SQL statement is extracted as the final answer.

LearNAT – Result

- **Main result: Results of Decomposition Synthesis Procedure**

Table 1: Results of *Decomposition Synthesis Procedure*. The decomposition success rate and token consumption on BIRD-train are reported.

Methods	Success Rate	Token Cost
CoT	59.07%	16,735K
MCTS	71.55%	334,694K
+ AST Guide	78.01%	133,877K
+ Self-improvement Demonstration		
(1 round)	79.33%	137,456K
(2 round)	79.73%	142,017K
(3 round)	80.00%	145,977K

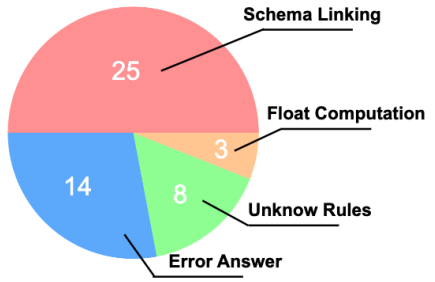


Figure 3: Error distributions of *Decomposition Synthesis Procedure* on randomly selected 50 error cases from the BIRD-train.

- 제안한 Decomposition Synthesis Procedure로 생성된 합성 데이터셋의 신뢰성을 확인하기 위한 실험 진행

- Self-improvement Demonstration 적용까지 한 버전은 80%의 decomposition 성공률을 보임

LearNAT – Result

- **Main result: Competition on the Public Leaderboard**

Table 2: Performance comparison on Spider-dev and Bird-dev benchmarks. All baseline results are taken directly from the performance reported on Leaderboard. **Bold** indicates the best result, while underline denotes the second-best results achieved by LearNAT.

Methods	*Venue*	*LLMs*	*BIRD-dev (In-Domain)*				*Spider-dev (Out-of-Domain)*				
			Simple	*Moderate*	*Challenging*	*Total*	*Easy*	*Medium*	*Hard*	*Extra Hard*	*Total*
System-Level											
C3-SQL		GPT-4	58.9	38.5	31.9	50.2	92.7	85.2	77.6	62.0	82.0
DIN-SQL	NeruIPS'23	GPT-4				50.7	91.1	79.8	64.9	43.4	74.2
MetaSQL	ICDE'24	GPT-4					91.1	74.7	64.1	36.1	69.6
MAG-SQL		GPT-4	65.9	46.2	41.0	57.6					85.3
SuperSQL	VLDB'24	GPT-4	66.9	46.5	43.8	58.5	94.4	91.3	83.3	68.7	87.0
MAC-SQL	COLING'25	GPT-4	65.7	52.7	40.3	59.4					86.7
Model-Level											
ACT-SQL	EMNLP'23	GPT-4					91.1	79.4	67.8	44.0	74.5
CatSQL	VLDB'23	N/A					95.8	88.3	74.7	62.7	83.7
DAIL-SQL	VLDB'24	GPT-4	62.5	43.2	37.5	54.3	90.3	81.8	66.1	50.6	76.2
SENSE	ACL'24	CodeLLaMA-13B				55.5	95.2	88.6	75.9	60.3	83.5
CodeS	SIGMOD'24	CodeS-7B	64.6	46.9	40.3	57.0	94.8	91.0	75.3	66.9	85.4
		CodeS-15B	65.8	48.8	42.4	58.5	95.6	90.4	78.2	61.4	84.9
Ours											
LearNAT		Qwen2.5-Coder-7B	65.4	48.4	42.4	58.1	95.2	**92.4**	76.4	67.5	86.4
		Qwen2.5-Coder-14B	68.5	51.4	45.8	61.2	95.6	91.5	80.5	68.7	86.9
		Qwen2.5-Coder-32B	**70.7**	**55.5**	**59.0**	**65.0**	**96.4**	**92.4**	**85.1**	**69.3**	**88.4**

- 제안한 LearNAT이 7b variant만으로도 System-level baseline에 근접한 성능을 도출, 32B variant는 SOTA 성능 달성
- System-level baseline은 pipeline 수준의 추가 접근법을 수행함에도 불구하고 LearNAT 대비 낮은 성능을 보인다는 점에서 큰 강점을 보임

LearNAT – Result

- **Main result: Comparison under Identical Evaluation Protocol**

Table 3: Performance comparison of `LearNAT` and competitive literature under identical evaluation protocol. **Bold** indicates the better result.

Methods	*Evaluation Protocol*	*LLMs*	*BIRD-dev (In-Domain)*				*Spider-dev (Out-of-Domain)*					*Total*
			Simple	*Moderate*	*Challenging*	*Total*	*Easy*	*Medium*	*Hard*	*Extra Hard*	*Total*	
SynCoT	SynCoT	Qwen2.5-7B-Instruct				59.2					78.9	67.1
`LearNAT`	SynCoT	Qwen2.5-Coder-7B	67.6	48.0	45.8	**59.6**	91.9	91.0	71.3	63.9	**83.6**	**69.2**
OmniSQL	OmniSQL	Qwen2.5-7B-Instruct				**63.9**					81.2	70.9
`LearNAT`	OmniSQL	Qwen2.5-7B-Instruct	68.2	50.3	50.7	61.1	96.0	91.5	77.6	65.1	**86.0**	**71.1**
SQL-o1	SQL-o1	Llama3-8B	71.8	52.3	45.2	63.4	94.4	93.0	81.0	68.7	87.4	73.1
`LearNAT`	SQL-o1	Qwen2.5-Coder-7B	**72.5**	**54.2**	**49.3**	**64.8**	**96.4**	**94.8**	**78.2**	**74.1**	**89.1**	**74.6**
Alpha-SQL	Alpha-SQL	Qwen2.5-Coder-7B	72.6	59.3	53.1	66.8	94.0	89.2	76.4	63.3	84.0	73.7
`LearNAT`	Alpha-SQL	Qwen2.5-Coder-7B	**74.4**	**61.5**	**52.8**	**68.4**	**97.2**	**96.0**	**80.5**	**77.1**	**90.6**	**77.4**

- Baseline별 inference 전략이 다르므로 **LearNAT의 checkpoint를 바탕으로 각 baseline별 Inference 전략을 적용했을 때의 성능 변화를 확인**하는 실험 진행
- OmniSQL의 BIRD-dev 성능 하락을 제외한 모든 경우에서 성능 향상이 이뤄짐. 이는 LearNAT이 제안하는 학습 방법의 LLM 자체의 Text2SQL 추론 능력 향상 및 확장성을 강조하는 실험에 해당

Reasoning Text2SQL

BIRD-INTERACT: RE-IMAGINING TEXT-TO-SQL EVALUATION VIA LENS OF DYNAMIC INTERACTIONS

Anonymous authors
Paper under double-blind review

ICLR 2026 oral

score: 8 / 8 / 8 / 6

Bird-Interact

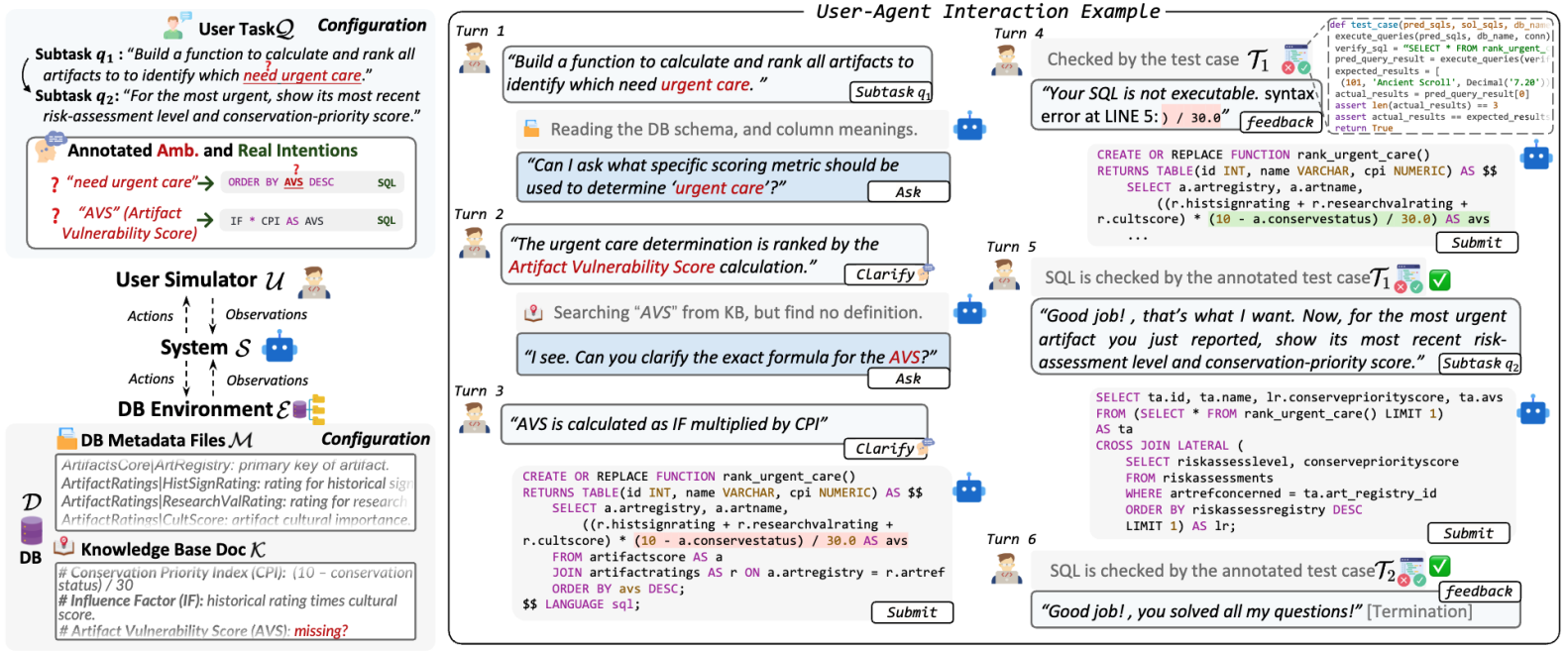


Figure 1: Task overview of BIRD-INTERACT showing the evaluated system interacting with DB Environment and User Simulator to complete the user task with a sequence of sub-tasks.

- 기존 Text2SQL 평가상황은 단 한번의 질의(single)로 해결할 수 있는 완전한 형태의 SQL 구문 생성(perfectly-formed)이 목표
- 그러나 현실세계에서 Text2SQL을 하는 상황은 모호한 질의를 다단계, 대화 형식으로 상호작용 하면서 사용자의 문제 상황을 해결해야함
  <- 이러한 현실세계의 Text2SQL 상황을 반영할 수 있는 Interactive Problem-solving process 모사 평가 플랫폼이 필요함

Bird-Interact – Method

- **Problem Formulation**

Multi-turn collaboration 상황에서 system의 Real-world Text2SQL 대응 능력을 평가하기 위한 문제 상황은 아래와 같음

- Text2SQL system S_θ은 DB Environment $\mathcal{E} = \{\mathcal{D}, \mathcal{M}, \mathcal{K}\}$를 바탕으로 User simulator $\mathcal{U}_\gamma$ 와의 Interaction을

$$u_i^t = \mathcal{U}_\gamma(h_i^{t-1}, q_i, \mathcal{E}), \quad s_i^t = \mathcal{S}_\theta(h_i^{t-1}, u_i^t, \mathcal{E}), \quad h_i^t = h_i^{t-1} \oplus \langle u_i^t, s_i^t \rangle \qquad (1)$$

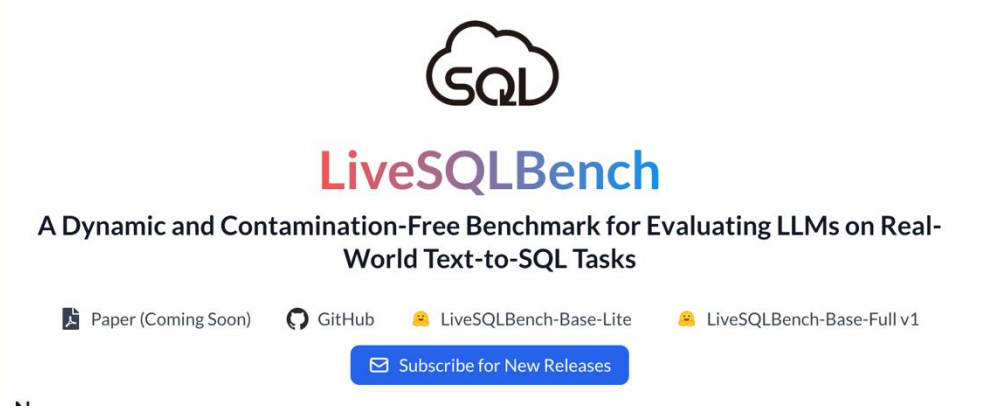
- i번째 Turn마다 user simulator는 history, i번째 sub-task q_i, DB Environment가 주어질 때 t번째 interaction을 생성
- i번째 Turn마다 system은 history, i번째 user query u_i^t, DB Environment가 주어질 때 t번째 답변을 생성
- 각 sub-task q_i마다 ground-truth SQL σ_i^*와 Test cases $\mathcal{T}_i$ mapping되어 sub-task 단위로 정확성을 확인할 수 있음
  - 각 task마다 2개의 sub-task로 분해될 수 있으며 ambiguity를 해소하기 위한 priority sub-task, follow-up sub-task로 구분될 수 있음

Bird-Interact – Method

- **Benchmark Construction**

LiveSQLBench 데이터셋을 기반으로 Benchmark 구축

- Read-only operation외에도 Data Manipulation Language, Data Definition Language 등 다양한 SQL case를 평가할 수 있는 평가 환경 조성
- benchmark 배포 및 재사용 가능
- Database의 metadata와 External Knowledge를 연계하는 Hierarchical Knowledge Base 존재
- 기존 LiveSQLBench는 single-turn benchmark로 본 논문은 이를 real-world 평가 환경을 반영할 수 있는 multi-turn benchmark를 구축하고자 함
- 12명의 전문 Annotator를 모집하여 ambiguity injection, follow-up sub-task generation을 통한 multi-turn benchmark로 변환을 실시



Rank / Window	Model	Organization	Success Rate (%) ↓	Avg. Cost (USD) / Task	Link
1 2024-11 – 2026-03	o3-mini	OpenAI	31.15	$ 0.0225	
2 2024-11 – 2026-03	GPT-5	OpenAI	31.15	$ 0.0383	
3 2024-11 – 2026-03	o4-mini	OpenAI	29.54	$ 0.0188	
4 2024-11 – 2026-03	o3	OpenAI	29.54	$ 0.1752	
5 2024-11 – 2026-03	Claude Sonnet 4	Anthropic	27.01	$ 0.0601	
6 2024-11 – 2026-03	Qwen3-235B-A22B	Qwen	26.90	$ 0.0043	

Bird-Interact – Method

- **Benchmark Construction**

1. Ambiguity Injection -> Priority sub-task generation

- 기존 user query에 superficial user query ambiguities, Knowledge ambiguities, Environmental ambiguities를 주입
  - superficial user query ambiguities는 user의 불분명한 단어 사용으로 ambiguity가 주입되는 경우 (intent-level ambiguities)와 범위, 시간 등 구현 디테일 부족으로 인한 ambiguity 주입 경우가 존재 implementation-level ambiguities
  - Knowledge ambiguities는 external knowledge $\mathcal{K}$에 특정 한개의 knowledge를 제거하여 ambiguity를 주입하는 one-shot knowledge ambiguity, multi-hop knowledge에서 중간 chain을 제거하는 knowledge chain breaking으로 ambiguity 주입
  - Environmental ambiguities는 기존 LiveSQLBench에서 metadata에 존재하는 ambiguity를 의미
- Ground-truth SQL에서 ambiguity가 주입된 부분에 해당하는 SQL snippet을 paring하여 clarification source로 사용

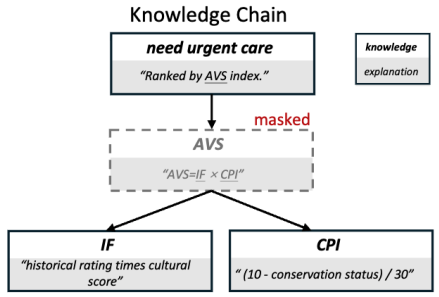


Figure 2: Knowledge chain breaking ambiguity.

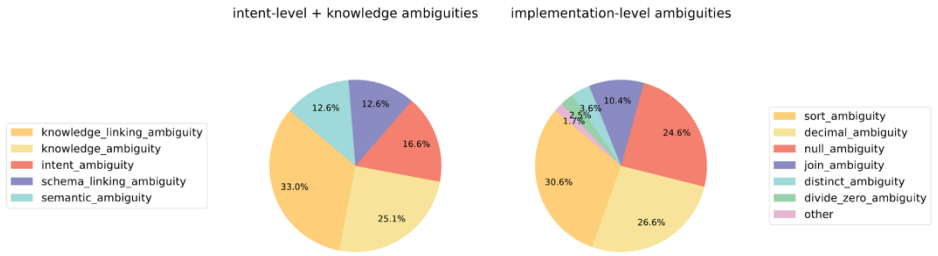


Figure 9: Ambiguity types distribution.

Bird-Interact – Method

- **Benchmark Construction**

2. Follow-up Sub-tasks Annotation -> Follow up sub-task generation

- Ambiguity injection으로 생성된 Priority sub-task에 pair되는 follow-up task를 5개 분류 체계에 따라서 생성
- 이러한 5개 분류체계는 Priority sub-task와 Follow up sub-task의 state dependency에 의거하여 지정

Table 8: Follow-up Sub-Task Taxonomy in BIRD-INTERACT

Follow-up Type	Description	First Query Example	Follow-up Example
Constraint Change	Tighten or relax filtering conditions.	*"List employees hired in 2024."*	*"Only engineers."* / *"Include 2023 as well."*
Topic Pivot	Compare or switch entity values to explore alternatives.	*"Sales of Product A in 2023."*	*"What about Product B?"*
Attribute Change	Modify the requested attributes, metrics, or columns.	*"Departments with >50 staff."*	*"Give their average salary."*
Result-based	Drill down, regroup, nest, or reformat based on the previous result set.	*"List projects finished in 2023."*	*"For Apollo, show its budget."*
Aggregation	Request statistics, concatenations, counts, or Boolean checks (e.g., `AVG`, `STRING_AGG`, `MAX FILTER`, `ARRAY_AGG+LIMIT`, `EXISTS`). Final output is typically a scalar, single row, or compact table.	*"Show the top-10 artists by track count."*	*"Give me their names joined into a single comma-separated string."*
Object-Dependent	First query creates or modifies a database object (e.g., table, view), and the follow-up query operates on it.	*"Create a table of employees with salary above 100k."*	*"From that table, list only engineers."*

Bird-Interact – Method

- **Function-Driven User Simulator**

  Interactive Multi-turn evaluation과 안정적인 평가를 위하여 Two-stage strategy를 활용한 User simulator 운용을 실시
  - LLM as Semantic Parser: User Simulator는 System 응답 결과를 Parsing하여서 AMB, LOC, UNA와 같은 미리 지정된 type별 대응 전략을 선택
  - 선택된 대응전략을 바탕으로 User Simulator는 응답을 생성

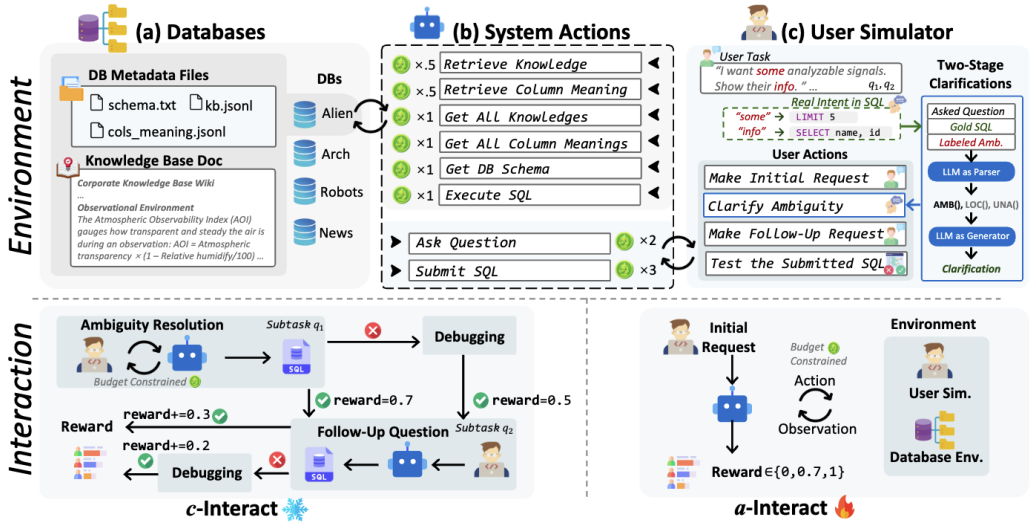


Figure 3: Two evaluation settings for BIRD-INTERACT: c-Interact, where the system engages in conversation with the user, and a-Interact, where the system interacts flexibly. At the end of the task, the system will receive a reward $r \in [0, 1]$.

Bird-Interact – Experimental Setups

- **Evaluation settings - c-Interact evaluation**

Multi-turn dialogue 방식으로 System과 User simulator가 interaction을 수행

- User는 초기에 ambiguous한 sub-question를 질문, System은 이에 관해 Ambiguity를 제거하기 위한 후속 질문 및 적절한 sub-query를 생성 해야함
  - System이 잘못된 SQL query를 주는 경우, 이를 해결할 수 있는 debugging 기회를 sub-question당 1회에 한하여 부여
- System 생성 sub-query는 dataset 구축 단계에서 tagging된 label을 바탕으로 correctness 확인, 후속 sub-question을 묻는 방식으로 대화를 이어감

Budget Constraints

$$\tau_{\text{clar}} = m_{\text{amb}} + \lambda_{\text{pat}}$$

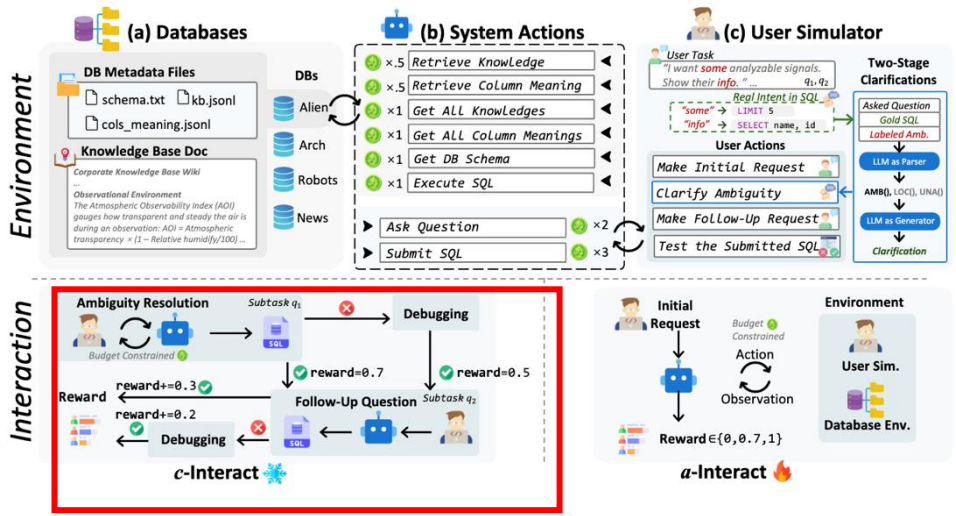


Figure 3: Two evaluation settings for BIRD-INTERACT: c-Interact, where the system engages in conversation with the user, and a-Interact, where the system interacts flexibly. At the end of the task, the system will receive a reward $r \in [0, 1]$.

Bird-Interact – Experimental Setups

- **Evaluation settings - a-Interact evaluation**

  Agent 평가 방식으로 사전 정의된 action space를 바탕으로 System이 필요에 따라서 user simulator와 interaction하는 평가 상황

  - Target database, metadata, HKB, User Simulator가 Callable Tool로 주어져 System은 초기에 주어지는 question을 해결하기 위한 Agentic한 방법으로 문제를 해결해야 함

Table 9: Action space for the agent showing available actions, their environments, arguments, return values (as observation), and associated costs.

Action	Env.	Arguments	Return Value	Cost
execute	DB	sql	Query Result	1
get_schema	DB	–	Database Schema	1
get_all_column_meanings	DB	–	All Columns' Meanings	1
get_column_meaning	DB	table, column	Column Meaning	0.5
get_all_external_knowledge_names	DB	–	All Knowledge Names	0.5
get_knowledge_definition	DB	knowledge	Knowledge Definition	0.5
get_all_knowledge_definitions	DB	–	All Knowledge Definitions	1
ask	User	question	User Clarification	2
submit	User	sql	User Feedback	3

Budget Constraints

$$B = B_{base} + 2\,m_{amb} + 2\,\lambda_{pat}$$

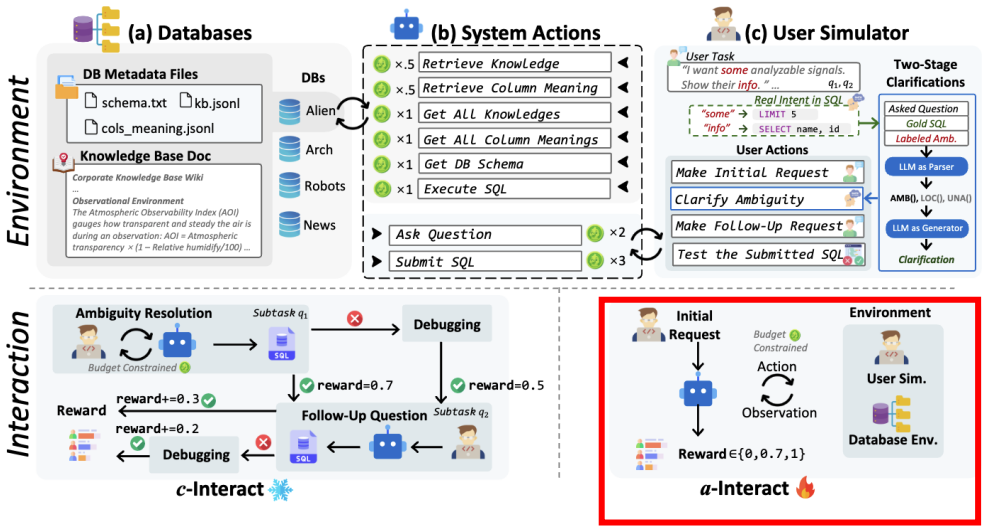


Figure 3: Two evaluation settings for BIRD-INTERACT: c-Interact, where the system engages in conversation with the user, and a-Interact, where the system interacts flexibly. At the end of the task, the system will receive a reward $r \in [0, 1]$.

Bird-Interact – Experimental Setups

- **Metric**

Success Rate (SR)

$$\mathbf{SR}_j = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left[\mathcal{T}_{i,j}(\sigma_{i,j}) = \texttt{True}\right],$$

Normalized Reward (NR)

$$R = \frac{\sum_i r_i}{N} = \frac{\sum_i \sum_{j\in\{1,2\}} r_{i,j}}{N},$$

$$r_{i,1} = \begin{cases} 0.7 & \text{if 1st sub-task is solved without debugging} \\ 0.5 & \text{if 1st sub-task is solved with debugging} \\ 0 & \text{otherwise} \end{cases}$$

$$r_{i,2} = \begin{cases} 0.3 & \text{if 2nd sub-task is solved without debugging} \\ 0.2 & \text{if 2nd sub-task is solved with debugging} \\ 0 & \text{otherwise} \end{cases}$$

$$r_i = \begin{cases} 1.0 & \text{if both sub-tasks are passed} \\ 0.7 & \text{if only the 1st sub-task is passed} \\ 0 & \text{otherwise} \end{cases}$$

Bird-Interact – Result

- **Main result**

Table 2: Success Rate and Final Normalized Reward of different models on BIRD-INTERACT-FULL. The success rate is cumulative; Reward* is the normalized reward (%). The values reported in *c*-Interact are after debugging phase, and (+n) means the performance gained via debugging. Avg. Cost is the cost for one task on average in USD. Our user simulator has an avg. cost of 0.03 USD.

Model	Priority Question (Success Rate %) ↑			Follow Ups (Success Rate %) ↑			Reward* ↑	Avg. Cost ↓
	BI	DM	Overall	BI	DM	Overall		
c-Interact Text-to-SQL								
GPT-5	9.49 (+0.00)	25.40 (+2.12)	14.50 (+0.67)	5.84 (+0.24)	14.81 (+0.53)	8.67 (+0.33)	12.58	$ 0.08
Claude-Sonnet-3.7	10.71 (+4.62)	33.86 (+7.41)	18.00 (+5.50)	4.62 (+0.49)	16.40 (+3.17)	8.33 (+1.33)	13.87	$ 0.29
Deepseek-Chat-V3.1	11.44 (+0.73)	33.86 (+3.17)	18.50 (+1.50)	4.62 (+0.24)	16.93 (+1.06)	8.50 (+0.50)	15.15	$ 0.12
Qwen-3-Coder-480B	16.30 (+2.68)	34.39 (+5.29)	22.00 (+3.50)	8.03 (+0.97)	16.93 (+4.23)	10.83 (+2.00)	17.75	$ 0.11
Claude-Sonnet-4	16.06 (+4.87)	35.98 (+10.58)	22.33 (+6.67)	10.46 (+1.22)	22.22 (+3.70)	14.17 (+2.00)	18.35	$ 0.29
O3-Mini	17.76 (+2.92)	37.57 (+11.11)	24.00 (+5.50)	11.44 (+0.73)	**25.40** (+4.23)	15.83 (+1.83)	20.27	$ 0.07
Gemini-2.5-Pro	**18.73** (+4.38)	**38.62** (+10.05)	**25.00** (+6.17)	**12.41** (+1.22)	24.87 (+5.29)	**16.33** (+2.50)	**20.92**	$ **0.04**
a-Interact Text-to-SQL								
Qwen-3-Coder-480B	8.05	24.74	13.33	3.90	4.74	4.17	10.58	$ 0.07
Deepseek-Chat-V3.1	10.49	31.58	17.17	4.63	5.26	4.83	13.47	$ **0.06**
O3-Mini	12.20	36.32	19.83	5.85	14.21	8.50	16.43	$ **0.06**
Gemini-2.5-Pro	10.49	41.58	20.33	5.85	20.00	10.33	17.33	$ 0.22
Claude-Sonnet-3.7	11.46	41.58	21.00	5.61	16.84	9.17	17.45	$ 0.60
Claude-Sonnet-4	15.85	53.68	27.83	8.05	22.63	12.67	23.28	$ 0.51
GPT-5	**15.61**	**58.42**	**29.17**	**10.98**	**30.00**	**17.00**	**25.52**	$ 0.24

- 5개의 close-source 모델 및 2개의 open-source model을 바탕으로 평가 진행
- GPT-5, Gemini-2.5-pro 모델도 20% 수준의 매우 낮은 SR, NR을 보임
  - c-interact, a-interact 상황에서 두 모델은 최소 16.33, 17의 SR을 기록

- Priority Question 보다 Follow-ups의 성공률이 현저히 낮음.
  이는 대화 History 가중에 따른 Context Length 제약으로 인한 결과로 추정

- 모델들은 복잡한 도메인 지식과 분석 추론이 필요한 비즈니스 인텔리전스(BI) 쿼리가 정형화된 패턴을 가진 데이터 관리(DM) 태스크보다 성능을 내기 어려웠음

- 모델별로 강점을 가지는 Interaction 상황이 다름을 알 수 있음.
  특히 GPT-5의 경우 C-interact 대비 a-interact에서 보다 SR이 높아지는 것을 알 수 있음

Bird-Interact – Result

- **Interaction Analysis – Memory Grafting**

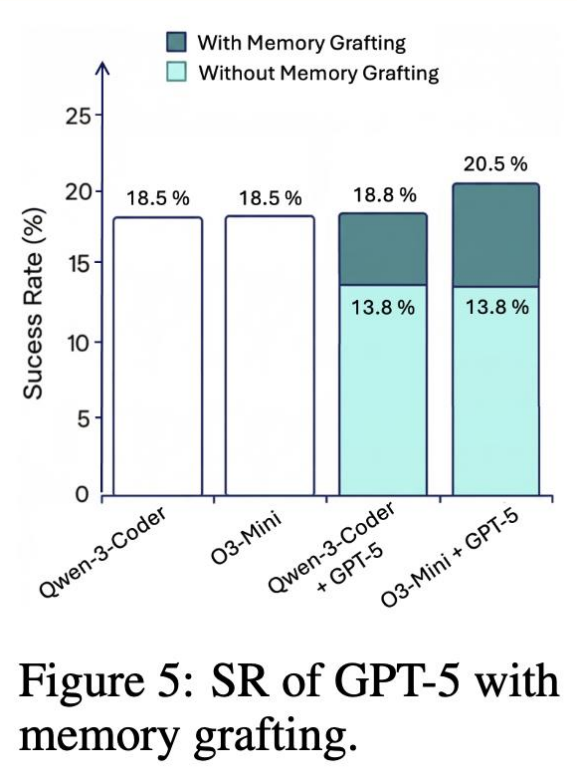


Figure 5: SR of GPT-5 with memory grafting.

- GPT-5의 c-interact 상황에서의 낮은 성능이 대화 상황에 대응 능력이 떨어지는 것에 기인하여 추가 실험을 진행

- Memory Grafting: 상대적으로 대화 능력이 뛰어난 Qwen-3-Coder, O3-Mini의 ambiguity resolution 기록을 GPT-5가 활용하여 c-interact 상황에서 문제를 해결하도록 설정

- 두 모델의 interaction history를 활용할 때 GPT-5의 성능이 개선되는 것을 확인

Bird-Interact – Result

- **Interaction Analysis – Interaction Test-Time Scaling**

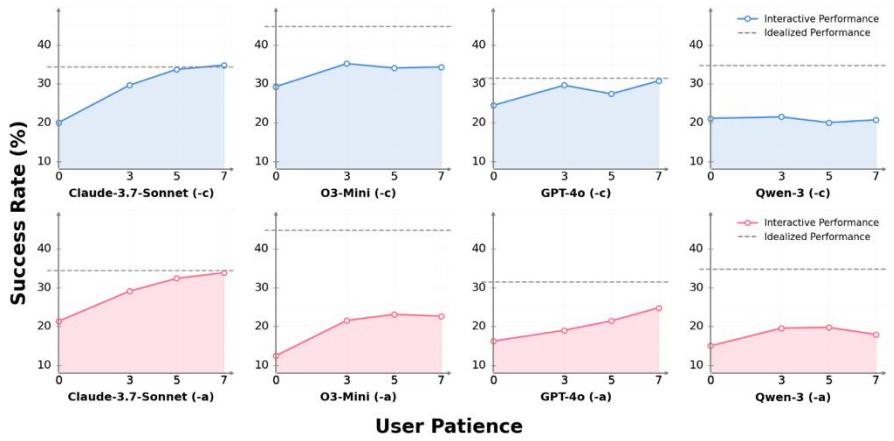


Figure 4: The performance of different LLMs with different user patience on BIRD-INTERACT-LITE. The red line denotes a-Interact mode (-a); the blue line denotes c-Interact mode (-c). And the dotted line (***Idealized*** Performance) denotes the performance under ambiguity-free single-turn text-to-SQL.

- 모델에게 주어지는 질문 및 대화 기회 허용 횟수(User patience)를 증가함에 따른 성능 변화를 확인

- Claude-3.7-Sonnet이 scailing 효과를 잘 받는 것을 확인. 많은 interaction이 허용될 경우 Oracle setting을 넘어설 수 있는 가능성을 확인

- 타 모델도 비슷한 양상을 보이나 Oracle setting 성능에 미치지 못함을 확인

Bird-Interact – Result

- **User Simulator Analysis**

Table 3: Correlation analysis between AI and human users.

User Simulator	Pearson (p-value)
GPT-4o - *w/ Func. (Ours)*	0.84 ($p = 0.02$)
Gemini-2.0-Flash - *w/ Func. (Ours)*	0.79 ($p = 0.03$)
GPT-4o - *Baseline*	0.61 ($p = 0.14$)
Gemini-2.0-Flash - *Baseline*	0.54 ($p = 0.21$)

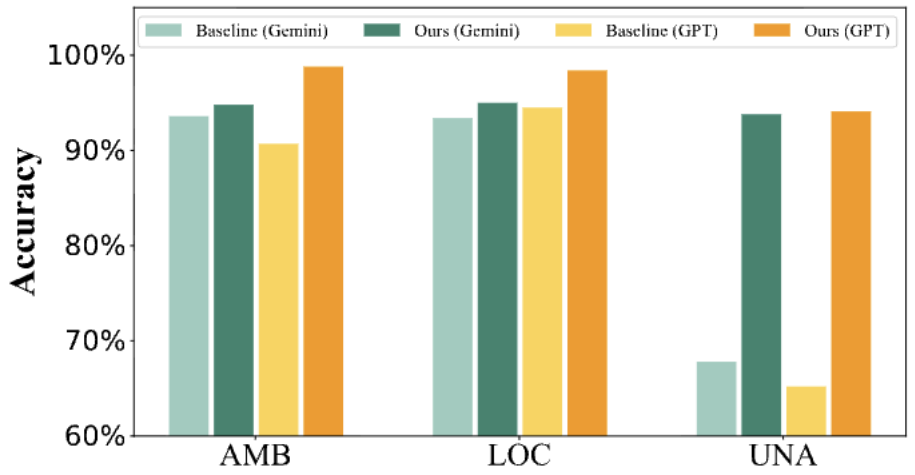


Figure 6: The accuracy of different user simulators on USERSIM-GUARD.

Applying Text2SQL for inter-research

Harnessing Temporal Databases for Systematic Evaluation of Factual Time-Sensitive Question-Answering in LLMs (ICLR 2026)

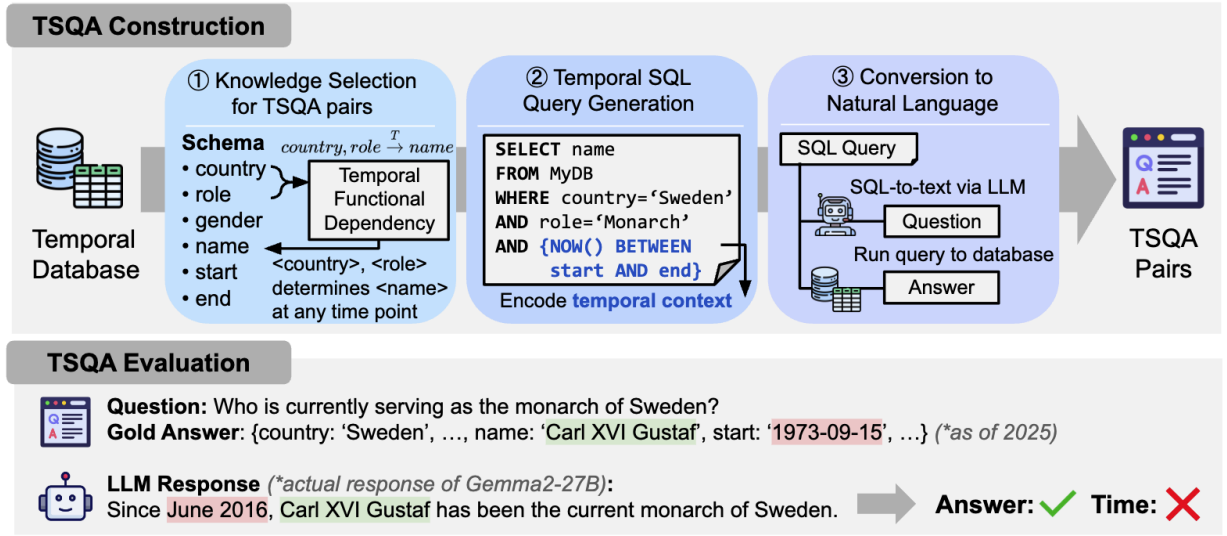


Figure 1: Overview of TDBench framework. TDBench systematically constructs Time-Sensitive QA (TSQA) pairs by (1) selecting factual knowledge via temporal functional dependencies, (2) generating temporal SQL queries with diverse temporal contexts, and (3) converting queries into natural language QA pairs using an LLM and the database. During evaluation, TDBench automatically verifies both the final answer and time references in LLM responses, capturing cases where the model hallucinates in the explanation despite providing the correct answer. TDBench supports diverse TSQA scenarios, including temporal alignment and temporal reasoning tasks – see more framework details in Sec. 3.

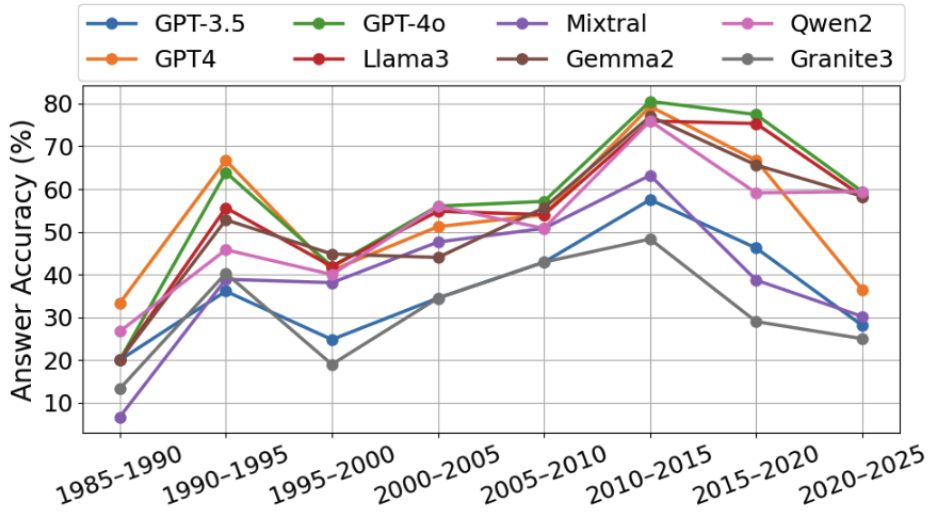


Figure 3: LLM performances across different time spans (1985-2025) in the closed-book setting. Additional results on multiple domains are presented in Sec. D.3.

Thank you